PATENT COOPERATION TREATY

PCT

INTERNATIONAL PRELIMINARY EXAMINATION REPORT

(PCT Article 36 and Rule 70)

Applicant's or agent's file reference	**FOR FURTHER ACTION** See Notification of Transmittal of International Preliminary Examination Report (Form PCT/IPEA/416)	
International application No. PCT/GB 03/02778	International filing date *(day/month/year)* 30.06.2003	Priority date *(day/month/year)* 28.06.2002
International Patent Classification (IPC) or both national classification and IPC G06F9/38		
Applicant CRITICAL BLUE LTD et al.		

1. This international preliminary examination report has been prepared by this International Preliminary Examining Authority and is transmitted to the applicant according to Article 36.

2. This REPORT consists of a total of 5 sheets, including this cover sheet.

☒ This report is also accompanied by ANNEXES, i.e. sheets of the description, claims and/or drawings which have been amended and are the basis for this report and/or sheets containing rectifications made before this Authority (see Rule 70.16 and Section 607 of the Administrative Instructions under the PCT).

These annexes consist of a total of 7 sheets.

3. This report contains indications relating to the following items:

- I ☒ Basis of the opinion
- II ☐ Priority
- III ☐ Non-establishment of opinion with regard to novelty, inventive step and industrial applicability
- IV ☐ Lack of unity of invention
- V ☒ Reasoned statement under Rule 66.2(a)(ii) with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement
- VI ☐ Certain documents cited
- VII ☐ Certain defects in the international application
- VIII ☐ Certain observations on the international application

Date of submission of the demand	Date of completion of this report
12.01.2004	02.03.2005
Name and mailing address of the international preliminary examining authority: European Patent Office D-80298 Munich Tel. +49 89 2399 - 0 Tx: 523656 epmu d Fax: +49 89 2399 - 4465	Authorized Officer Thibaudeau, J Telephone No. +49 89 2399-2349

I. Basis of the report

1. With regard to the **elements** of the international application *(Replacement sheets which have been furnished to the receiving Office in response to an invitation under Article 14 are referred to in this report as "originally filed" and are not annexed to this report since they do not contain amendments (Rules 70.16 and 70.17))*:

Description, Pages

1-3, 6-27	as originally filed
4, 5	received on 10.02.2005 with letter of 04.02.2005

Claims, Numbers

1-43	received on 10.02.2005 with letter of 04.02.2005

Drawings, Sheets

1-6	as originally filed

2. With regard to the **language**, all the elements marked above were available or furnished to this Authority in the language in which the international application was filed, unless otherwise indicated under this item.

   These elements were available or furnished to this Authority in the following language: , which is:

   ☐ the language of a translation furnished for the purposes of the international search (under Rule 23.1(b)).

   ☐ the language of publication of the international application (under Rule 48.3(b)).

   ☐ the language of a translation furnished for the purposes of international preliminary examination (under Rule 55.2 and/or 55.3).

3. With regard to any **nucleotide and/or amino acid sequence** disclosed in the international application, the international preliminary examination was carried out on the basis of the sequence listing:

   ☐ contained in the international application in written form.

   ☐ filed together with the international application in computer readable form.

   ☐ furnished subsequently to this Authority in written form.

   ☐ furnished subsequently to this Authority in computer readable form.

   ☐ The statement that the subsequently furnished written sequence listing does not go beyond the disclosure in the international application as filed has been furnished.

   ☐ The statement that the information recorded in computer readable form is identical to the written sequence listing has been furnished.

4. The amendments have resulted in the cancellation of:

   ☐ the description, pages:

   ☐ the claims, Nos.:

   ☐ the drawings, sheets:

5. ☐ This report has been established as if (some of) the amendments had not been made, since they have been considered to go beyond the disclosure as filed (Rule 70.2(c)).

   (Any replacement sheet containing such amendments must be referred to under item 1 and annexed to this report.)

6. Additional observations, if necessary:

V. Reasoned statement under Article 35(2) with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement

1. Statement

Novelty (N)	Yes:	Claims	1-43
	No:	Claims	
Inventive step (IS)	Yes:	Claims	1-43
	No:	Claims	
Industrial applicability (IA)	Yes:	Claims	1-43
	No:	Claims	

2. Citations and explanations

   see separate sheet

1. Reference is made to the following document:

   D3: US-A-5 819 067 (LYNCH THOMAS W) 6 October 1998 (1998-10-06)

2. **Item V : Reasoned statement under Article 35(2) with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement**

   D3, which is considered to be the closest prior art document with respect to the present wording of claim 1, discloses a computer system which includes at least two microprocessors. The first microprocessor is configured to translate instructions from an original computer program coded in a first instruction set to a translated computer program coded in a second instruction set. The second microprocessor is configured to execute the translated program. Instead of manually performing the translation, the translation is automatically performed by the computer system upon invocation of the original computer program. Those computer programs which may be more efficiently executed by the second microprocessor may be translated into the instruction set executed by the second microprocessor.
   The invention provides a method of configuring a microprocessor architecture comprising the steps of:
   - translating code executable on a first type of microprocessor into code executable on a second type of microprocessor, said second type having the microprocessor architecture to be configured;
   - automatically adapting at design time the instruction set of the microprocessor architecture to provide performance benefits for the second type of microprocessor when executing the translated code;
   - executing the translated code on the configured microprocessor.

   In D3, both processors are fixed architectures whose instruction sets do not change. D3 does not disclose the adaptation of the instruction set for the second microprocessor at design time in order to provide performance benefits for said second processor. Therefore, as far as claim 1 can be understood (see objection of lack of clarity below), its subject-matter seems to involve an inventive step.

3. **Item VII: Certain defects in the international application**

i) To meet the requirements of Rule 6.3(b) PCT the independent claims should have been properly cast in the two part form, with those features which in combination are part of the prior art (see document D3) being placed in the preamble.

ii)The features of the claims are not provided with reference signs placed in parentheses (Rule 6.2(b) PCT).

iii) Contrary to the requirements of Rule 5.1(a)(ii) PCT, the relevant background art disclosed in the document D3 is not mentioned in the description, nor is this document identified therein.

iv) According to the PCT Preliminary Examination Guidelines, III, 4.3a, relating to "spirit and scope", the material contained in the description on last page 27, last paragraph, is not clear (Article 6 PCT).

4. **Item VIII: Certain observations on the international application**

It is not clear what "adapting" ... "the instruction set" ... "to provide performance benefits" ... "in order to provide a configured processor" means. Claim 1 does not comprise in which way said adaptation should be performed in order to obtain said performance benefits.

Indeed it is definitely the wish of every skilled person knowing D3 to provide performance benefits to its second processor. Merely mentioning "adapting at design time the instruction set of the microprocessor architecture" does not define the technical means necessary for achieving the desired result.

It is clear from the description on page 4 and on page 9 that the following features are essential to the performance of the invention:

- instruction set extension is supported by converting calls to particular software functions into an invocation of an extension hardware unit;
- a hardware unit is designed that performs the same operation as said particular software function.

Since independent claim 1 does not contain this feature it does not meet the requirement following from Article 6 PCT that any independent claim must contain all the technical features essential to the invention (see also the PCT Guidelines, PCT/GL/4 III-4.3).

SUMMARY OF THE INVENTION

This document discloses a method of configuring a microprocessor architecture, comprising the steps of:

(a) taking executable code that can execute on a first type of microprocessor and translating it such that the resultant translated code can execute on a second type of microprocessor, the second type of microprocessor having the microprocessor architecture which is to be configured;

(b) automatically adapting at design time the instruction set of the microprocessor architecture to provide performance benefits for the second type of microprocessor when executing the translated code in order to provide a configured microprocessor; and then

(c) executing the translated code on the configured microprocessor.

Microprocessors configured in this way are targeted at embedded systems applications that execute repetitive code that contains high degrees of potential parallelism. The microprocessors are configured and programmed automatically by the analysis of the application software in the form of an executable image in the ISA of a particular host microprocessor. The configured microprocessors have a customised target ISA that is specifically designed to exploit parallelism in the application software. The instruction set translation operates by converting each source machine instruction into a sequence of more basic operations to be executed on a target processor. All registers reads and writes are formed into separate operations. Thus a 3-address add operation is converted into operations to read the left and right operand registers, the add operation itself and finally an operation to write back the result to the register file. Instructions with complicated addressing modes or that modify the condition codes result in longer sequences of basic operations.

One disadvantage of prior art static translation systems is that they are unable to support debugging using the host instruction set. It is obviously imperative to support existing debuggers. The innovations in the translation approach are primarily in the methods to allow such debug support. By maintaining a correspondence between the host processor state and the coprocessor state at specified points in the execution it is possible to support host level breakpoints on the architecture. In other words a breakpoint can be set specified by a host

instruction address and the architectural state reproduced as though the code was actually running on the host processor.

Instruction set extension is supported by converting calls to particular software functions into an invocation of an extension hardware unit. A hardware unit is designed that performs the same operation as a particular software function. That is, it takes the same parameters and produces the same results as the code in its software equivalent. The advantage of the hardware version is that it will be able to produce the results more quickly.

The configured target microprocessors may be used as coprocessors within a system. They are responsible for executing certain software functions translated from an executable image of another host microprocessor. This host microprocessor will typically also be present in the system. Mechanisms are provided to allow the host microprocessor and target coprocessors to interact and maintain coherence between the memory systems.

CLAIMS

1. A method of configuring a microprocessor architecture, comprising the steps of:
(a) taking executable code that can execute on a first type of microprocessor and translating it such that the resultant translated code can execute on a second type of microprocessor, the second type of microprocessor having the microprocessor architecture which is to be configured;
(b) automatically adapting at design time the instruction set of the microprocessor architecture to provide performance benefits for the second type of microprocessor when executing the translated code in order to provide a configured microprocessor; and then
(c) executing the translated code on the configured microprocessor.

2. The method according to claim 1 whereby the configured microprocessor architecture is used in a coprocessor in a system.

3. The method according to claim 2 whereby a host processor in the system is of the type of the executable used to configure the coprocessor.

4. The method according to claim 3 whereby the host processor in a system runs the executable used to configure the coprocessor.

5. The method according to claim 4 whereby a number of individual software functions in the executable code are marked for translation and execution on the coprocessor.

6. The method according to claim 5 whereby an original executable image is automatically modified so that function calls to those translated functions cause an equivalent function to be executed on the coprocessor.

7. The method according to claim 6 whereby a coprocessor initiation involves the transfer of register state from the host processor to the coprocessor.

8. The method according to claim 7 whereby completion of a function on the coprocessor causes the transfer of register state from the coprocessor to the host processor.

9. The method according to claim 7 whereby completion of a function on the coprocessor causes the transfer of memory state from the coprocessor to the host processor.

10. The method according to claim 1 whereby the configured architecture generated is designed to execute parts of the executable with higher performance than can be achieved with a host processor.

11. The method according to claim 10 whereby the higher performance is obtained by the execution of more operations in parallel than is achieved with the host processor.

12. The method according to claim 1 whereby the architecture generated is designed to execute parts of the executable with lower power consumption than can be achieved with a host processor.

13. The method according to claim 1 whereby the executable code is translated into the instruction set of the configured processor.

14. The method according to claim 13 whereby each instruction in an executable image is translated into one or more basic operations.

15 The method according to claim 14 whereby each of these operations may be performed by a particular execution unit that is present in the configured processor.

16. The method according to claim 15 whereby a register file is present as an execution unit in the architecture and explicit operations to read and write the register file is generated as part of the translation.

17. The method according to claim 16 whereby static register analysis may be used to eliminate unnecessary writes of registers.

18. The method according to claim 17 whereby code is subdivided into atomically executed blocks.

19. The method according to claim 18 whereby each atomically executed block reproduces the operations of the corresponding host code.

20. The method according to claim 19 whereby the state of live registers at the end of the atomic block execution is identical to that obtained from execution on the host processor.

21. The method according to claim 19 whereby the state of the memory at the end of the atomic block execution is identical to that obtained from execution on the host processor.

22. The method according to claim 1 whereby breakpoints may be set on the configured architecture.

23. The method according to claim 22 whereby the breakpoints may be specified using the addresses of instructions in the original executable.

24. The method according to claim 23 whereby the nearest preceding instruction for which state can be synchronised on the configured processor is determined when the breakpoint is set.

25. The method according to claim 24 whereby the configured processor contains a mechanism to determine the equivalent target instruction address for a host instruction address.

26. The method according to claim 25 whereby the configured processor contains hardware to cause a breakpoint halt on the required address that prevents any side effects caused by sequentially later instructions.

27. The method according to claim 26 whereby upon detection of a breakpoint execution can be continued on the host processor from the synchronised address until the point of the actual breakpoint.

28. The method according to claim 23 whereby the breakpoint address is determined by decoding the data stream on the debug interface to the host processor.

29. The method according to claim 1 whereby certain host processor instruction addresses may be converted to target processor addresses while the system is running.

30. The method according to claim 29 whereby a hashing table is maintained in memory to perform a mapping of certain host processor instruction addresses to target addresses.

31. The method according to claim 30 whereby mapping information may be interleaved with the machine code for the target processor.

32. The method according to claim 31 whereby the state of certain bits within each word of a table are used to indicate whether the information represents an address mapping entry or target machine code.

33. The method according to claim 1 whereby function calls in the executable code may be replaced with uses of particular hardware blocks in the configured processor.

34. The method according to claim 33 whereby the input parameters to the software function correspond to the operands supplied to the corresponding hardware unit.

35. The method according to claim 34 whereby the reference parameters and return result from a software function correspond to the results generated by the corresponding hardware unit.

36. The method according to claim 33 whereby the original software implementation may be used as a behavioural model for hardware for the purposes of simulation.

37. The method according to claim 1 whereby an instruction set translator converts instructions from an executable image into behaviourally equivalent operations that are mapped to a target processor.

38. The method according to claim 4 whereby the coprocessor contains one or more cache memories.

39. The method according to claim 38 whereby the coprocessor and host processor communicate via a system bus or a generic coprocessor interface on the host processor.

40. The method according to claim 39 whereby the host processor services memory access requests from the coprocessor while the coprocessor is operational.

41. The method according to claim 40 whereby the coprocessor is able to flush its caches of all modified data when the end of function execution is reached.

42. The method according to claim 39 whereby a copy of some virtual to physical page mappings are maintained by the coprocessor.

43. A microprocessor that has been automatically configured using the method as defined in any preceding claim 1 – 42.